Thursday, 6 May 2021 8:31 PM

Z - Algorithm

→ Naive → $(P * T)$
→ Rabin-Karp → $(P * T)$ → (T)

→ Core of the algorithm is the function
→ $Z(k)$ = longest substring starting at (k) which is also prefix of the string.

string → a a b x a a y a a b
Z → 0 1 0 0 2 1 0 3 1 0

substring at index (1), length 1 is a prefix of this string.

1. To use this information for pattern matching
2. How to compute the z-array efficiently.

Q. text → x abc abz abc ✓
pattern → abc

(1) Concat pattern + text and separate them by a character which is not present in any of these strings.

abc $ x abc abz abc

z box

(2) Calculate the z-array of this new concatenated string.

→ a b c $ x a b c a b z a b c   len(P)
0 1 2 3 4 5 6 7 8 9 10 11
Z → 0 0 0 0 0 3 0 0 2 0 0 3 0 0

(3) look for positions where length of the pattern is equal to the z-array element.

To find the original index in text where pattern is present,

index − (length of pattern + special char)
5 − (3 + 1) = 1
11 − (3 + 1) = 7

original array indices.

Time Complexity → $O(|P| + |T|)$

length of text; length of pattern

$P * T$

Quick ques

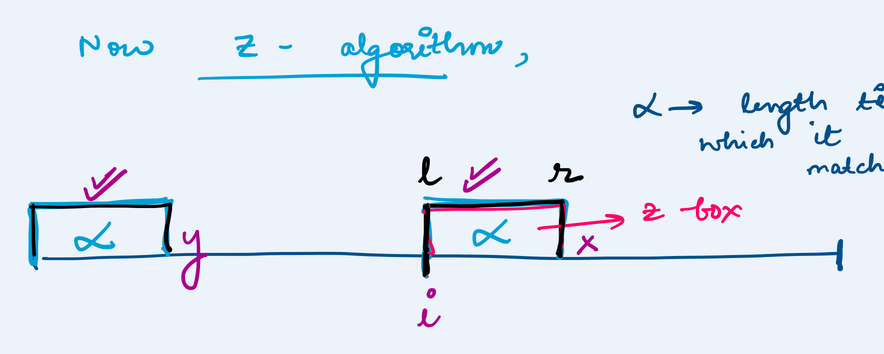
$S = P \$ T$

⇒ $\boxed{S = T \$ P}$?? (No)

This is not a symmetric property.

Now z-algorithm,

α → length till which it matches



$Z_i > 0$

Now, given at position i, where $Z_i > 0$,
⇒ starting at position i, there is some substrings that matches the prefix of the string

$\boxed{y \neq x}$

string of length |S|
i = 0 → i = |S|
compute |z| from i = 1 to i = |S|

(1) Compute Z_1

a a a a b x

z 0 3

a a
a a
a a
a b

for just computing Z_i = 4 comparisions

worst case ??

a a a a a

Z_1 compute → Z_1

Z_1 → 4

l = 5

comparisions → 5

(1) Naive approach → very ineffective.

k1 k1 l k

a a b x a a b x c a a b x a a b x a y
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Z → 0 1 0 0 4 1 0 0 0 8 1 0 0 5 1 0 0 1 0

19 ≈ (28)

comparisions → 𝍸 𝍸 𝍸 𝍸 𝍸 |||

When copying z-values, if the sum of z-value + current index ≥ right boundary, of z-box, then we need to re-calculate the z-value.

Q1 Given two string A and B, count how many cyclic permutations of B when taken XOR with A gives 0.

XOR property

Approach: XOR of a no with a same no is 0.

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

A ^ A = 0

⇒ So, we have to count how many times substring of (B) matches with string (A).

→ Concatenate B with B, so that it contains all cyclic permutations.

10110

→ Now the problem is reduced to finding the occurances of pattern A in string S = B·B

→ Z - algo.

101 $ 10110

001 0 3 0 1 2 0

ans → (1)

Q. Given a string A of size N. consisting of uppercase alphabets. The operation allowed is to insert characters in the $\boxed{\text{beginning}}$ of the string.

* Find the return how many characters are needed to make the string a palindrome string.

Eg (A) → "ABC" output → (2)

Insert B → BABC
Insert C → CBABC

Approach (1) abc TC → $O(n^2)$

Keep removing the last char until you either find the string palindromic or empty.

output → (2)

Approach (2)

$\boxed{\text{string \$ reverse string}}$

✓ string → "aa cec a aaa"

aacecaa a a $ a a aacecaa a

Z → 0 1 0 0 0 2 2 2 1 0 2 2 (7) 1 0 0 0 2 1

→ find the max val of the array.

9 − 7 = (2)

reversed string xx

a a a a a c e c a a $ a a c e c a a a a a

Z → 0 3 2 1 0 0 0 2 1 0 2 1 0 0 0 4 3 2 1

Q. Given a very large number n, count the total no. of ways such that if we divide the numbers into two parts A and B, then no. a can be obtained by integral division of number B by some power p of 10, and p ≥ 0

Eg: 220
output → 1
Explanation
→ 2 20
A = 2 B = 20 p = 1
20/10^1 = 2

Eg: 2202200
output → 2
Explanation

(1) 2 2 0 2 2 0 0
A = 2 B = 20200
p = 5
$\frac{20200}{10^5} = 2.02$

✓ (2) 2 2 0 2 2 0 0
A = 220 B = 2200 (p = 1)
$\frac{2200}{10}$

(1) Irregular division of no is not allowed
289 → 29 and 8 xx

(2) a and b should not contain leading zeroes.

(3) Integral division of no is allowed F(a/b)

(4) 1 ≤ no. of digits in N ≤ 10^5